EXPERIMENTS IN AGGREGATING AIR ORDINANCE EFFECTIVENESS DATA FOR THE TACWAR MODEL

THESIS

James E. Parker
Major, USAF

AFIT/GOA/ENS/97M-12

DEPARTMENT OF THE AIR FORCE
AIR UNIVERSITY
AIR FORCE INSTITUTE OF TECHNOLOGY

Wright-Patterson Air Force Base, Ohio

AFIT/GOA/ENS/97M-12

EXPERIMENTS IN AGGREGATING AIR ORDINANCE EFFECTIVENESS DATA FOR THE TACWAR MODEL

THESIS

James E. Parker
Major, USAF

AFIT/GOA/ENS/97M-12

AFIT/GOA/ENS/97M-12

EXPERIMENTS IN AGGREGATING AIR ORDINANCE EFFECTIVENESS DATA FOR THE TACWAR MODEL

THESIS

Submitted to the Faculty of the Graduate School of Engineering
in Partial Completion of the
Requirements for the Degree of
Master of Science in Operations Research

James E. Parker, BSCE
Major, USAF

February, 1997

THESIS APPROVAL

Student: James E. Parker, Major, USAF Class: GOA-97M

Title: Experiments in Aggregating Air Ordinance Effectiveness Data for the TACWAR Model

Defense Date: 20 February 1997

Committee:	Name/Title/Department	Signature
Advisor	Jack Jackson Assistant Professor Department of Operational Sciences	[illegible]
Reader	Steven K. Rogers Professor Department of Electrical and Computer Engineering	[illegible]

Preface

The purpose of this research effort was to develop a straightforward method of aggregating output data from the SABSEL model for use in the TACWAR model. The objective was to develop a software application that would provide consistent results while not requiring a tremendous amount of effort on the part of the analyst. The majority of the effort focused on the construction of a functional approximation of the data and evaluation of this approximation against a look-up table methodology. For the SABSEL data set, a look-up table proved superior. An application was built using the look-up features of MS Access that is simple and straightforward.

This work could not have been completed without the help and encouragement of others. First , I am most indebted to my Lord and Savior, Jesus Christ without whose sacrifice none these efforts would have the slightest eternal meaning. I would also like to thank my advisor, Ltc Jack Jackson, for his patient encouragement and pointed comments that helped maintain focus on the big picture. I'm also grateful to Dr. Steve Rogers, who helped keep this effort on track with his insight and direction throughout my efforts with neural networks. Major Don Hinton also provided great help with his insights into the vagaries of MS Access and saved me many hours of trial and error. Finally, I would like to give special thanks to my wife, Michelle, whose consistent love, patience and support were invaluable.

Table of Contents

Page

List of Figures

List of Tables

AFIT/GOR/ENS/97M-12

Abstract

An interactive MS Access™ based application that aggregates the output of the SABSEL model for input into the TACWAR model is developed. The application was developed following efforts to create a functional approximation of the SABSEL data using neural networks, statistical networks, and traditional statistical techniques. These approximations were compared to a look-up table methodology on the basis of accuracy, (RMSE < 0.01), speed (runs in minutes) and compactness (storage requirement). The method best satisfying these criteria, the look-up table, was used as the basis from which the application was constructed. Results from the comparison of the function approximation techniques give insight as to the strengths and weaknesses of each technique.

EXPERIMENTS IN AGGREGATING AIR ORDINANCE EFFECTIVENESS DATA FOR THE TACWAR MODEL

I. Introduction

A recurring challenge in modeling and simulation (M&S) is the aggregation of data from high resolution models for input or use in lower resolution models. For example, the U. S. Central Command Combat Analysis Group (CCCA) often uses output from the Air Force's Sabre/Selector (SABSEL) model for input into the TACWAR theater level model, and needs an analytically sound method to aggregate the SABSEL data. This research focuses primarily on the construction of a functional approximation of SABSEL data using neural networks, statistical networks, and traditional statistical techniques. The approximations constructed under each technique will be compared with a look-up table methodology on the basis of accuracy, speed and compactness (storage requirement). Finally, using the best methodology, an automated aggregation tool for the SABSEL data will built to meet CCCA's needs.

1.1 Background

The CCCA uses theater level combat models for much of its analysis. Many inputs to these models require the aggregation of high resolution data into a useable form for input to one of these models. Currently, much of the aggregation of data relies upon the expert judgment of the analyst to boil down high resolution model data. Due to variability in methodology from analyst to analyst, this expert judgment produces results that are hard to defend when subjected to skeptical scrutiny. As the reliance of decision makers upon M&S continues to increase, the current method of aggregation is losing its utility. A better method is needed.

More specifically, CCCA has a need for an analytically sound method of aggregating the air ordinance effectiveness data of the engineering level SABSEL model for input of a single value into the TACWAR theater level model. The large SABSEL data base contains values for numerous factors including aircraft type, ordinance load, delivery profiles, target type and layout, weather, and many others. From these many factors SABSEL can produce output for a wide variety of input scenarios. The challenge for CCCA analysts is producing a consistent, analytically defensible input value to the TACWAR model in a timely fashion.

The difficulty of this aggregation process can be illuminated with the following example. Currently, TACWAR has a single effectiveness value for an AGM-65 missile attacking a single tank ground target. SABSEL on the other hand generates separate values for several AGM-65 missiles, delivered from several different aircraft, various weather conditions, differing delivery tactics, differing target types, differing target groupings, the number of weapons on of the delivery aircraft, the number of passes over the target and numerous other factors. Each of these factors may or may not have a distinct effect on the effectiveness of a particular combination of input values. Currently, CCCA uses an expert judgment based estimation process. In this process the analyst averages the SABSEL effectiveness data for those factors the analyst deems significant. Since the heuristic varies from analyst to analyst, this leads to a lack of defensible consistency in the aggregation process. CCCA desires a methodology that is straightforward, transparent, and defensible (Foulk, 1996).

1.2 Research Objectives

This research will compare several methods of creating a functional approximation as a means of aggregating the data from the SABSEL model. This effort will focus on building and training a neural network to produce the desired output. The performance of the neural network will be compared to an automated off the shelf software product that combines statistical and neural networks techniques. The output of both of these techniques will also be compared to a statistical regression approach. This comparison will be based on the accuracy of the functional approximation using the root mean square error (RMSE) due to the ability to compare the fit of various sample sizes. For those techniques achieving the desired accuracy, considerations of speed and compactness will be compared to that of a look-up table based approach. Using the "best" technique, an automated tool was created for aggregating the SABSEL data.

1.3 Research Approach

In recent years, problems of this nature have been tackled by network modeling approaches that produce output based on a polynomial representation of the various input factor levels. One of the advantages of these networks is that they are self organizing. Self organizing models can "train" on input data and adjust the output values to provide a desired result in a manner similar to how a child learning a new skill makes adjustments to produce the desired result (Nelson, 1991:3). This research will first design and train a neural network using a portion of the SABSEL data. Upon achieving consistent results with the neural network, a second model was built using the automated modeling tool ModelQuest^{TM} by AbTech Corporation of Charlottesville, VA (ModelQuest, 1996).

Then, a third model will be built using traditional statistical techniques. These models will then be measured against a look-up table for speed and accuracy. Accuracy will be measured against root mean square error (RMSE) using a portion of the SABSEL data set held out of the model building process. The "best" method will be that which most closely conforms to CCCA's requirements of accuracy, clarity, and defensibility.

A successful research effort will produce insight into the various significant factors thereby enhancing the aggregation process for CCCA. Our goal is to produce a straightforward, analytically sound methodology that CCCA can apply consistently. As a best case, this research will produce or demonstrate a methodology that has wide applicability for many aggregation problems facing the M&S community.

Chapter Two discusses the applicable literature outlining the techniques used to approximate the SABSEL data as a function. The techniques discussed include backpropagation and radial basis function neural networks, statistical networks, and principal components analysis. Chapter Three provides a narrative of the research and results, to include the building of a MS Access application to meet the sponsor's requirements. Chapter Four outlines the final results and recommendations from this research effort.

1.4 Summary

This thesis will apply neural networks, statistical networks, and traditional statistical techniques to the SABSEL database to attempt to generate a highly accurate function approximation of the data. The technique yielding the "best" performance will be measured against a look-up methodology and an application to aggregate the SABSEL data will be built to meet CCCA's requirements.

II. Literature Review

This problem is essentially one of approximating an unknown function which represents the underlying trends in the data, with a requirement for a high degree of accuracy. CCCA's current methodology uses a table look-up from the SABSEL data base to determine a single shot probability of damage (SSPD) as input to the TACWAR model. Neural nets, abductive networks, and multivariate statistics each show promise for reproducing the SABSEL data in the form of a function. This chapter provides a review of literature concerning neural networks, statistical networks, and multivariate analysis as each pertains to this problem. Specifically, the necessary terms for each technique will be defined and a basic discussion provided.

2.1 Neural Networks

Artificial neural networks grew out of attempts to model the processes of the brain at the level of the neuron. Neural network techniques attempt to reproduce biological capabilities by using a similar approach. Though biologically inspired, many of the methods in artificial neural networks are not biologically accurate in a manner similar to the way aircraft wings bear only faint resemblance to the birds wings which inspired them (Nelson, 1991:1) (Rogers, 1990:iii). Additionally, neural networks have grown to encompass a wide area of application using a variety of techniques. This discussion will focus on feedforward, multilayered, neural networks to perform a curve fitting operation on the multidimensional SABSEL data. The author assumes a familiarity with the terms of neural network research. However, a brief glossary is provided in appendix A for clarification.

2.1.1 Multilayer Perceptrons

The basic building block of the multilayer perceptron (MLP) is the single perceptron(see glossary, Appendix A). Linearly separable classes can be separated using a single perceptron. The perceptron is trained to return an output value of ≈ 1 with one class and an output of ≈ 0 with the other class. Through training the weights are adjusted based on the error between the actual output and the desired output. In cases where the data is not linearly separable but hyperplane separable, an MLP can be used to separate the classes (Rogers, 1996:7). Cybenko demonstrated that a single hidden layer MLP with enough perceptrons can approximate any continuous nonlinear function with arbitrary accuracy (Cybenko, 1989). Using a MLP for function approximation essentially forms a piecewise approximation to the function, with each perceptron training to return a segment of the function.

As in other curve fitting techniques, the problems of overfitting and underfitting the data present themselves. As a result, the selection of the number of perceptrons for fitting an unknown function remains an art form based on the practitioner's experience and *a-priori* knowledge of the data. Figure 1, depicts a Multilayer Perceptron with a single output. Though multiple outputs are feasible, the nature of the data for this study requires only a single perceptron on the output layer.

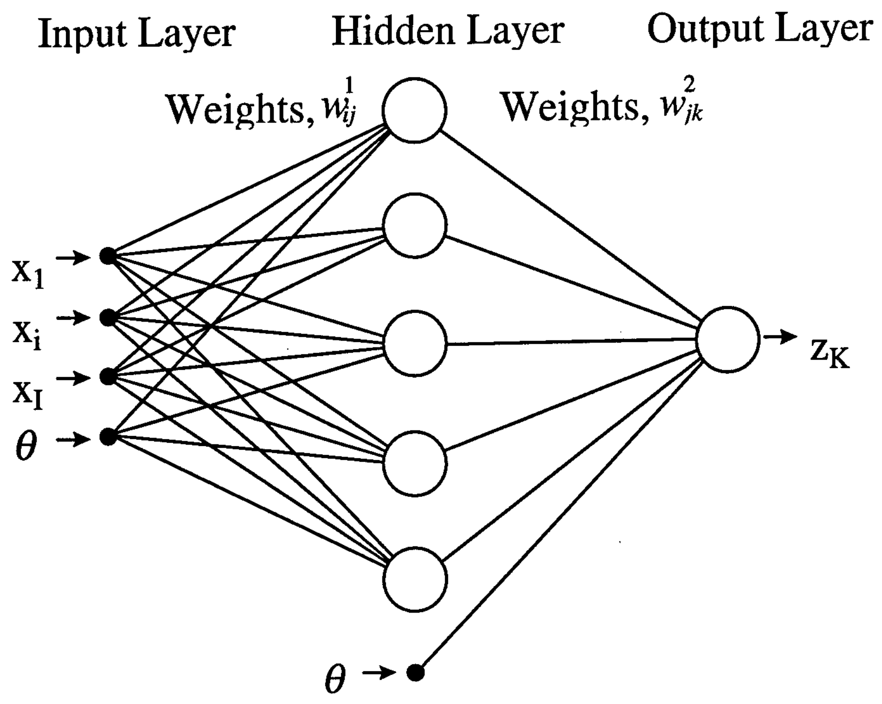


Figure 1. Multilayer Perceptron

2.1.1.1 Backpropagation Learning Law

The most common method for updating the matrix of weights of an MLP is backpropagation. This technique uses the partial derivative of the error, E, computed with respect to each weight, to calculate the updates. A common measure of error is sum squared error (SSE) defined as:

$$E = \sum_{i=1}^{K} (d_k - z_k)^2, \tag{1}$$

where z_k is the actual output and d_k is the desired output. In the case of a function approximation, only a single output perceptron is used and the subscripts of equation 1 are not necessary.

Let **w** be the weight matrix for a layer. Using gradient descent, the direction of steepest descent can be defined by $\frac{\partial E}{\partial \mathbf{w}}$. This relationship yields a learning law for updating the weights of each perceptron:

$$\mathbf{w}^h(t+1) = \mathbf{w}^h(t) - \eta \frac{\partial E}{\partial \mathbf{w}^h(t)}, \qquad (2)$$

where **w**(t+1) is the set of updated weights at step t+1, **w**(t) the old set of weights at step t, h defines the perceptron layer, and η is the learning rate or step size for the descent (Rogers, 1996:12-13). As written, this learning law is independent of the transfer function.

The process of training a MLP has four basic steps (Rogers, 1990:51).

1. Initialize the network weights. Rogers recommends using small random numbers with a uniform distribution between -0.5 and 0.5.
2. Input the training vector (exemplar) and desired output for that training vector to the network.
3. Calculate the actual output by using the outputs of each layer as inputs to the following layer (inputs for the first layer are the feature values of an exemplar):

   $$y_j = f(\sum_{i=0}^{I} w_i x_i + w_{I+1}\theta), \forall i = 1,2,\ldots,I, j = 1,2,\ldots,J, \qquad (3)$$

   where x_i are the inputs to the layer, w_i are the weights, y_j is the output of the each perceptron, θ is the bias, and $f(\bullet)$ is the transfer function.

   In matrix terms equation 3 is:

$$y_j = f(\mathbf{w}^T\mathbf{x}), \tag{4}$$

where the bias, θ, is included as the last row of the input vector **x**.

4. Training:

$$\mathbf{w}^h(t+1) = \mathbf{w}^h(t) - \eta \frac{\partial E}{\partial \mathbf{w}^h(t)}. \tag{5}$$

The key to the learning law and hence, backpropagation, is calculating the partial derivative of the error, E, with respect to the weights. This calculation is relatively straight forward provided the transfer function, $f(\bullet)$, has a manageable derivative. Transfer functions which have relatively simple derivatives include the sigmoid function (or logistic), the hyperbolic tangent, the linear function. These functions are shown in figure 2. Weight update equations for these functions are listed in table 1.

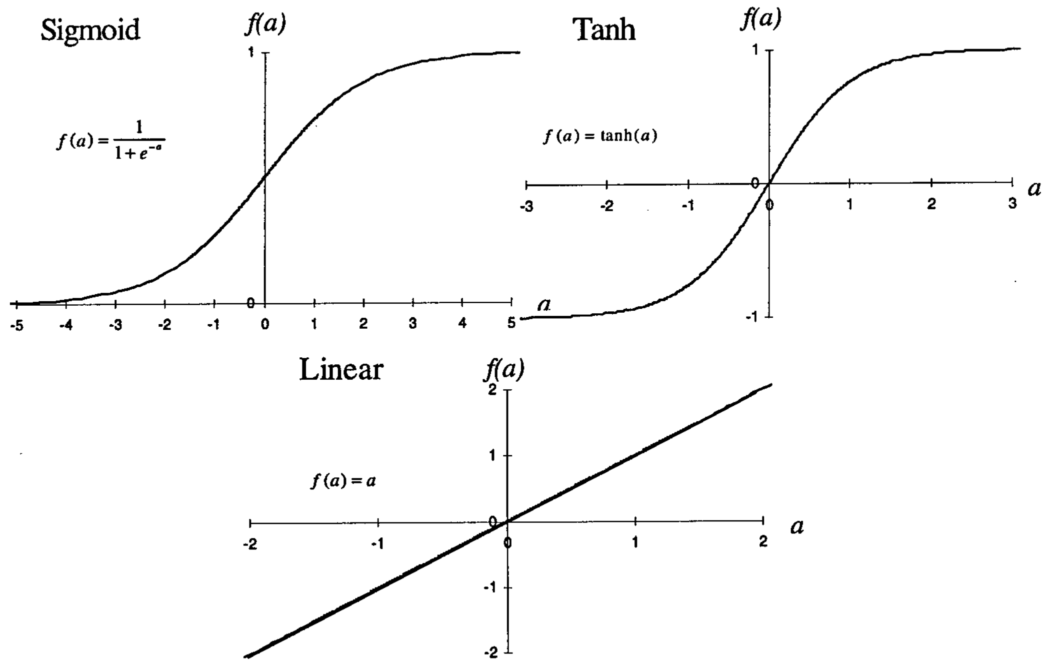


Figure 2. Transfer Functions

Table 1. Weight Updates

Layer	Transfer Function	Weight Update Equation
Output	Sigmoid	$w_j^2(t+1) = w_j^2(t) + \eta(d-z)(z)(1-z)(y_j)$
Output	Linear	$w_j^2(t+1) = w_j^2(t) + \eta(d-z)(y_j)$
Hidden	Tanh(linear output)	$w_{ij}^1(t+1) = w_{ij}^1(t) + \eta(d-z)(w_j^2)(1-(y_j)^2)(x_i)$
Hidden	Tanh(sigmoid output)	$w_{ij}^1(t+1) = w_{ij}^1(t) + \eta(d-z)(z)(1-z)(w_j^2)(1-(y_j)^2)(x_i)$

2.1.1.2 Enhancements to the Backpropagation Learning Law

Backpropagation, like other gradient descent techniques, tends to hang up on a local minimum rather than a global minimum. Two techniques that may speed learning and reduce the probability of hang up in a local minimum are momentum and adaptive step size. Momentum aids convergence on the global minimum by preventing radical changes in the weight change direction. Adding momentum to equation 4 results in:

$$\mathbf{w}^h(t+1) = \mathbf{w}^h(t) - \eta\frac{\partial E}{\partial \mathbf{w}^h(t)} + \mu(\mathbf{w}^h(t) - \mathbf{w}^h(t-1)), \tag{6}$$

where μ is the momentum constant. Empirical evidence suggests a value of μ is between 0.5 and 0.9 (Demuth, 1992). Another technique that speeds convergence or learning is adaptive adjustment of the step size or learning rate. If the learning rate is too small, convergence may be extremely slow. If the learning rate is too large, the weights may not converge as the weights oscillate between less than optimal solutions. An adaptive learning rate increases the step size if the error, E, decreases and decreases the step size if the error, E, increases. Adaptive learning can be implemented as follows:

$$\begin{array}{ll} \eta(t+1) = \eta(t) - \alpha\eta(t), & \mathit{if}\,(E(t) - E(t-1)) \le 0, \\ \eta(t+1) = \eta(t) + \alpha\eta(t), & \mathit{otherwise} \end{array} , \tag{7}$$

where α is the rate of change in the step size. Adding adaptive learning to equation 5 results in:

$$\mathbf{w}^h(t+1) = \mathbf{w}^h(t) - \eta(t+1)\frac{\partial E}{\partial \mathbf{w}^h(t)} + \mu(\mathbf{w}^h(t) - \mathbf{w}^h(t-1)). \tag{8}$$

Another commonly used adaptive technique consists of the logarithmic decay of the learning rate with each epoch (Rogers, Nov 1996).

An additional enhancement to the gradient descent is an adaptive technique that transitions between standard first order steepest descent and second order Gauss-Newton descent is the Levenberg-Marquardt algorithm (LM) (Neter, 1996:546). LM seeks to capitalize on the best features of both techniques Gauss-Newton and steepest descent. Gauss-Newton is a direct search procedure that converges quickly provided the starting conditions are sufficiently close. With poor starting conditions, Gauss-Newton may converge slowly, converge to a local minimum, or diverge (Neter, 1996:546). LM transitions between steepest descent and Gauss-Newton based on the rate of convergence of each epoch [E(t)-E(t-1)]. Typically LM converges orders of magnitude faster than the backpropagation techniques outlined above, however it requires significantly more computer memory (Demuth, 1992).

2.1.2 Radial Basis Functions

A large disadvantage of the preceding backpropagation techniques is that the gradient search techniques are computationally intensive and may not reach a global minimum. A viable alternative is the use of a network of radial basis functions (RBF). The key advantage is that this network is linear in the parameters and can be solved using the least squares algorithm.

As initially proposed, this technique centered an RBF such as a Gaussian or spline function on each data exemplar and solved for the weights. This produced an approximation with zero error, but did not prove practical in large applications. Various authors proposed techniques to reduce the size of the RBF network such that $I_s << I$, where I is the number of exemplars. These efforts met with varying degrees of success. Chen, et. al., proposed using a forward regression technique using orthogonal least squares(OLS) that selects a suitable set of RBF centers from the exemplars to obtain a parsimonious network as shown in appendix B (Chen, 1990:302). The Matlab neural networks toolbox implements the OLS algorithm (Demuth, 1992)

2.2 Statistical Networks™ by ModelQuest™

ModelQuest is an automated, self organizing software package that fits a piecewise polynomial to the input data. The ModelQuest system forms this fit by forming a network of piecewise polynomials where each node represents a fractional portion of the data similar to neural network techniques. ModelQuest constructs a high order network from the input data vectors where the output of each layer is a weighted algebraic sum of the inputs as shown in figure 3. The names single, doublet and triplet are based on the number of inputs to the element. The output of a single is determined by:

$$w_0 + (w_1 x_1) + (w_2 x_1^2) + (w_3 x_1^3) \tag{9}$$

where w_i are the weights, and x_i are the layer inputs. The output of a double is:

$$\begin{aligned} w_0 + (w_1 x_1) + (w_2 x_2) + (w_3 x_1^2) + (w_4 x_2^2) + (w_5 x_1 x_2) + \\ (w_6 x_1^3) + (w_7 x_2^3) + (w_8 x_2 x_1^2) + (w_9 x_1 x_2^2) \end{aligned} \tag{10}$$

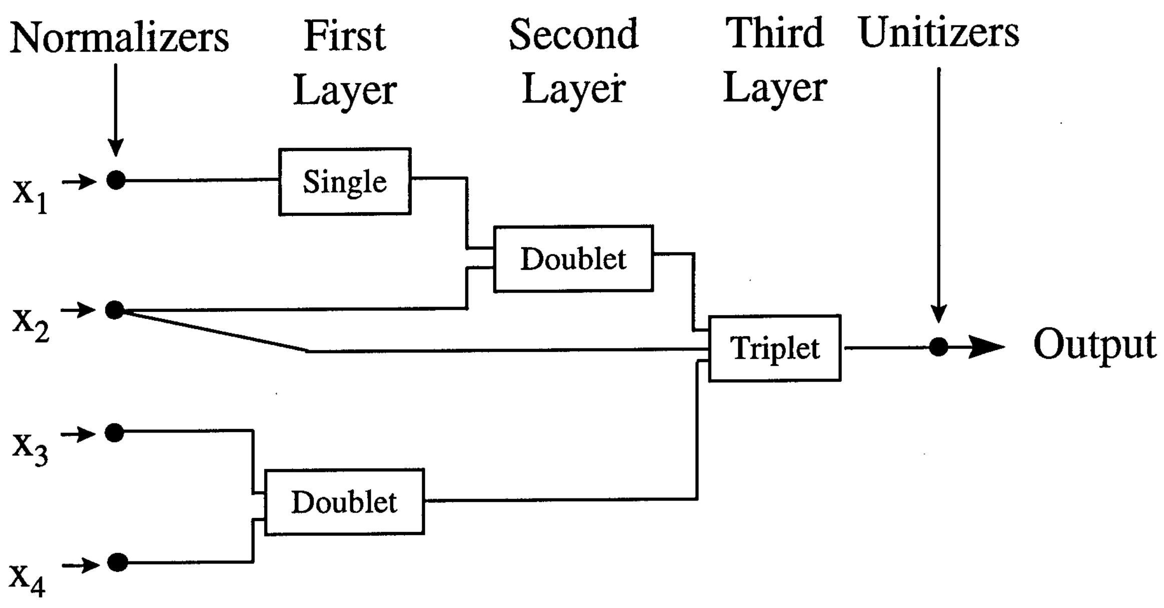


Figure 3. Example Four Input, Three Layer, Single Output Statistical Network

The output of a triple is determined by:

$$w_0 + (w_1 x_1) + (w_2 x_2) + (w_3 x_3) + (w_4 x_1^2) + (w_5 x_2^2) + (w_6 x_3^2) + (w_7 x_1 x_2) + (w_8 x_1 x_3) + (w_9 x_2 x_3) + (w_{10} x_1 x_2 x_3) + (w_{11} x_1^3) + (w_{12} x_2^3) + (w_{13} x_3^3) + (w_{14} x_2 x_1^2) + (w_{15} x_1 x_2^2) + (w_{16} x_1 x_3^2) + (w_{17} x_3 x_1^2) + (w_{18} x_3 x_2^2) + (w_{19} x_2 x_3^2) \tag{11}$$

Note that each element is a third degree polynomial and includes cross products in the case of doublets and triplets. Another element used by the network is the linear element such that the output is the weighted sum of all outputs from the preceding layer:

$$w_1 x_1 + w_2 x_2 + \text{K}\ w_n x_n \tag{12}$$

where n is the number of inputs from the preceding layer. Additionally, the network normalizes the input variables (features) with a zero mean and one variance, and unitizes the output variables to the same range (mean and variance) as the desired output.

ModelQuest automatically fits the best network using a criterion defined by A. R. Barron called predicted square error (PSE) (Barron: 1984). This criterion attempts to find the best balance between accuracy and complexity and is defined as follows:

$$\mathbf{PSE = SSE + KP} \tag{13}$$

where SSE is the sum squared error for the fitted model and KP is a complexity penalty. The complexity penalty KP can be adjusted based on the needs of the user. The model performance relationship between SSE, PSE and KP is shown in figure 4.

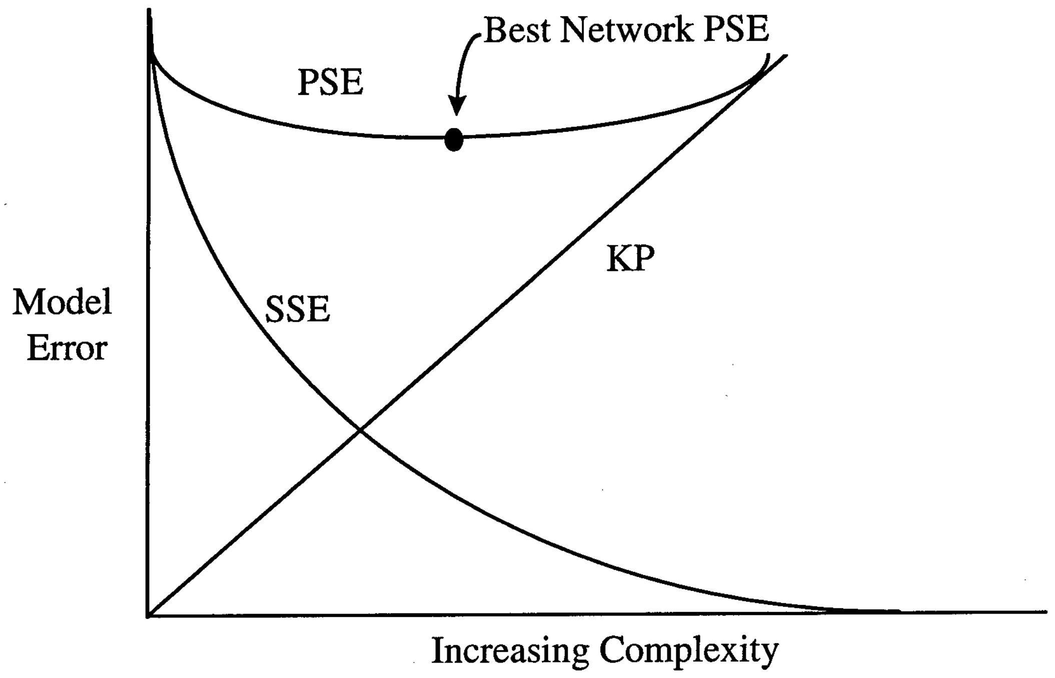


Figure 4. ModelQuest Performance

ModelQuest conducts an enumerative search of the possible combinations of linear, single, doublet and triplet elements against a training set of data to find the network which returns the best (lowest) PSE. The network is then presented against a test set of the data to check for the quality of generalization that may be attributed to the network (ModelQuest, 1996:2.6-2.8).

2.3 Statistical Methods

There are a myriad of statistical tools and techniques that could be brought to bear on this problem. The primary techniques that show promise for this problem are least squares regression, stepwise regression (JMP, 1995:199-210), the use of scatter plots (Neter, 1996:217-251), sorting and grouping (JMP, 1995), and principal component analysis (Bauer, 1996) (JMP, 1995). Non-linear regression (Neter, 1996:531-558) was considered as a means of providing a more accurate function approximation, but rejected because it requires gradient descent techniques for solution, and requires a high degree of practical skill to produce acceptable results, while not having the time saving advantage of self organization possessed by neural networks. Logistic regression was considered but rejected due to the nature of the desired output. Logistic regression works well when the output is qualitative, while the desired response for this research is continuous (Neter, 1996: 534). Therefore, no attempts at functional approximation will be made using logistic regression. A brief discussion of principal components analysis follows.

Principal component analysis (PCA) assumes an underlying structure within the data and seeks to determine the true dimensionality of a data set by studying the interdependence structure of a set of variables and provide a means of interpreting the underlying factors within the data (Bauer, 1996:15). By solving the correlation matrix (or covariance matrix) for its eigenvalues and normalized eigenvectors, the original features are mapped onto "principal components" as shown in figure 5. This technique takes "linear combinations of the original features such that the first principal component has maximum variation, the second principal component has the next most variation subject to being orthogonal to the first and so on"(JMP, 1995:315).

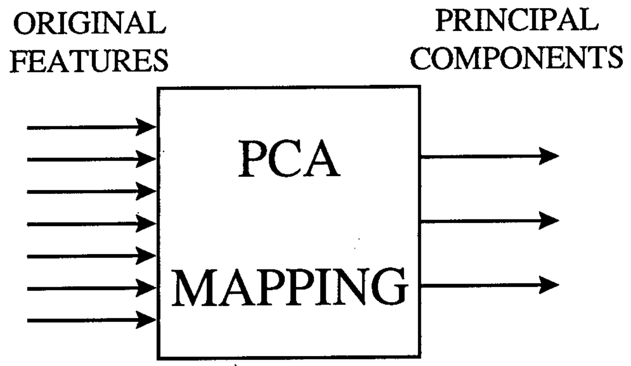


Figure 5. Principal Components Mapping

This mapping forms a linear combination as the cross product of the vector of original features and the eigenvector corresponding to each eigenvalue. PCA reduces the dimensionality by using only those principal components which explain a significant amount of the total variance within the data set (Bauer, 1996:15-26).

2.4 Summary

A function approximation of the SABEL data will be constructed using neural networks, statistical networks and statistical regression techniques. Each of these methods will be compared against a table look-up methodology, with the best in terms of accuracy, speed and simplicity serving as an engine for the desired aggregation tool.

III. Methodology and Results

The primary objective of this thesis was to develop a straightforward methodology for aggregating SABSEL data for use in the TACWAR model. Ideally, the solution could be implemented in common software such as Microsoft Excel™ or Access™ and would be rather transparent to the user. Most of the research focused on attempts to create a highly accurate function approximation of the SABSEL data that would execute efficiently. This approximation would serve as an "engine" for an MS Excel based application. Efficient in this situation, would be an approximation that executes quickly, requires significantly less memory than the look-up table methodology currently being used, and sacrifices little in terms of accuracy. Due to the nature of the SABSEL data, the accuracy turned out to be the driver behind most of the efforts. The following provides a narrative of the steps taken to create the desired function approximation. Though each technique approaches this problem from a different direction, there were areas in which the methods complemented each other. Finally, an application was built using MS Access to provide the desired aggregation tool. In chapter four, a recommended research methodology will be discussed.

3.1 Neural Networks

3.1.1 Gaining Experience with Backpropagation

The research began with an effort to code a feedforward, backpropagation neural network using the Matlab neural network toolbox. Using examples from the Matlab toolbox, this proved to be a relatively straight forward task. This code was tested on a

SUNSPARC Ultra computer against the following equation for verification and to gain experience with the various backpropagation techniques:

$$y = \left(\frac{1}{x+1}\right)\left[\frac{2}{1+\exp\{\sin(6x-\sqrt{5x})\}}\right] \tag{14}$$

which results in the decreasing sin wave type function that goes to zero as the input value goes to infinity shown in figure 6.

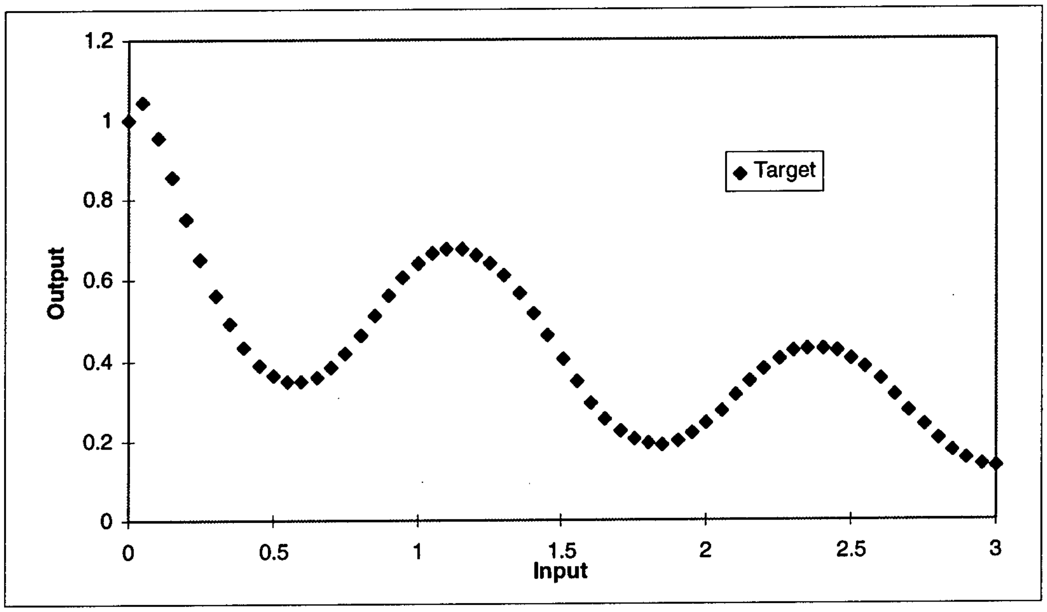


Figure 6. Data for Function Approximation

The measure of accuracy was set at 0.001 using sum squared error as the criterion. Experimentation was conducted using backpropagation, backpropagation with momentum, adaptive backpropagation and the Levenberg-Marquardt (LM) algorithm. For this two dimensional problem with a relatively small number of exemplars, no network required more than 5 minutes to train. The number of epochs required to train to the desired accuracy varied with the technique used and the number of perceptrons (or nodes) used. As stated in the Matlab neural networks toolbox manual, LM requires magnitudes of order fewer epochs to train. LM also demonstrated better characteristics in avoiding a local minimum. Additionally in conjunction with the LM algorithm, the number of nodes in the hidden layer were varied to demonstrate the effects of overfitting and underfitting. Too many nodes resulted in a jagged approximation, while too few nodes prevented reaching the error goal. One insight gained from this process is that LM will train quickly initially, then follow a relatively shallow slope to the desired error goal, unless there are too few nodes in the hidden layer.

3.1.2 Backpropagation Efforts with the SABSEL Data Set

The previously generated code served as the basis for the attempts to train a neural network against the SABSEL data (see Appendix C). The desired performance goal was the approximation of Single Shot Probability of Damage (SSPD) to a root mean square error (RMSE) less than 0.01. RMSE can be defined as:

$$RMSE = \sqrt{\frac{SSE}{n}}, \qquad (15)$$

where n is the number of exemplars (input vectors). A simple inspection of the data set showed that many of the features were a function of SSPD rather than independent variables. For example, 'expected kills' is the product of SSPD, the number of passes over the target and the number of weapons released per pass. Elimination of these variables reduced the feature space to twelve. A network was then trained on these features with no preprocessing of the data. Normally preprocessing consists of scaling or normalizing the data with a mean of zero and a standard deviation of one. The number of nodes were varied from ten to fifty.

During this initial attempt it became apparent that the size of the data base would greatly slow training, and would require considerable amounts of processor time. To increase neural network performance, while providing a source of data with which to check for the ability of the network to generalize, the data was split into equal sized training and test sets by selecting alternate exemplars for each set from the original data set. Additionally, a draw back of the LM algorithm soon became apparent. The LM algorithm requires a large amount of random access memory (RAM) to execute. This is due to the size of the matrix that must be inverted to solve the LM algorithm during each epoch. Once the available 64 megabytes of RAM of the machine were exhausted, the learning process proceeded at a snail's pace due to the slow file swapping process. With twelve features, network training slowed considerably when the number of nodes used exceeded fifty. Training on the entire data set and using 50 nodes produced a RMSE of 0.24 after 10000 epochs, which took over 150 hours of processor time. The desired accuracy was orders of magnitude larger than desired, and increasing the number of hidden layer nodes would increase the training time greatly, therefore additional reduction of features would be necessary.

3.1.2.1 Feature Reduction

The first attempt to improve the performance to the network involved using simple statistical techniques available through the JMP software to try and eliminate features from future training. Exploration of the data first involved generating a correlation matrix to help eliminate features that were highly correlated and discover any features that were highly correlated to the desired output SSPD. The correlation matrix in table 2 shows that the feature showing the highest correlation with SSPD is the weapon type, however its value is quite low at 0.2985.

Table 2 Correlations

Variable	Aircraft	Weapon	Loadout	Profile	Weather	Target	Tgt elem	**SSPD**
Aircraft	1	0.1888	0.5088	0.2282	-0.4723	-0.0217	-0.0027	-0.062
Weapon	0.1888	1	-0.1809	0.2358	-0.1583	0.0003	-0.0071	**0.2985**
Loadout	0.5088	-0.1809	1	-0.003	-0.4043	0.0354	0.0111	-0.1713
Profile	0.2282	0.2358	-0.003	1	-0.1995	-0.0376	-0.0068	0.0311
Weather	-0.4723	-0.1583	-0.4043	-0.1995	1	-0.0281	-0.0021	0.1809
Target	-0.0217	0.0003	0.0354	-0.0376	-0.0281	1	-0.0509	0.0593
Tgt elem	-0.0027	-0.0071	0.0111	-0.0068	-0.0021	-0.0509	1	-0.0938
SSPD	-0.062	0.2985	-0.1713	0.0311	0.1809	0.0593	-0.0938	1

Principal component analysis (PCA) and experimentation with different groupings of features, while searching for those combinations which could account for the greatest amount of variance within the data, produced disappointing results. In following the rule of thumb for PCA which suggests keeping those components which have an eigenvalue greater than 1.0 only four principal components should be used. These components roughly correspond to aircraft, weapon, target elements, and target type as shown in table 3. These components explain only 76% of the variance, thus failing to meet the accuracy requirements.

However, the grouping of aircraft, weapon, profile and target produced a set of features that spanned the entire data set. In other words, each exemplar within the set is a unique combination of values from these four features. Using this information, all but these four features were eliminated from future neural network training attempts. Closer examination of these four features showed that they all are nominal variables: aircraft type, weapon type, profile type, and target type. This would suggest there exists no real structure within the data. The scatter plot analysis in figure 7 confirms the lack of structure and indicates a relatively large number of nodes will be required to approximate this data with an acceptable error.

Table 3 Principal Components

Eigenvalue:	2.0167	1.3015	1.0524	0.9499	0.7581	0.5512	0.3703
Percent:	28.8094	18.5925	15.0337	13.5704	10.8306	7.8737	5.2897
CumPercent:	28.8094	47.4019	62.4356	76.006	86.8366	94.7103	100
Eigenvectors:							
Aircraft	**0.59366**	-0.0276	0.03065	-0.0452	-0.1191	0.47867	-0.6329
Weapon	0.17487	**0.68202**	-0.0898	0.14154	-0.5648	0.17279	0.35683
Loadout	0.4846	-0.4956	0.0299	-0.0417	0.03863	0.25927	0.66946
Profile	0.28015	0.52313	0.02207	-0.0359	0.79373	0.0361	0.12151
Weather	-0.551	0.01477	0.03201	-0.0457	0.13217	0.81795	0.0811
Target	0.00911	-0.1178	-0.7064	**0.68142**	0.12989	0.05794	-0.0498
Tgt elem	0.00128	-0.0262	**0.69971**	0.71308	0.03116	0.01005	-0.0129

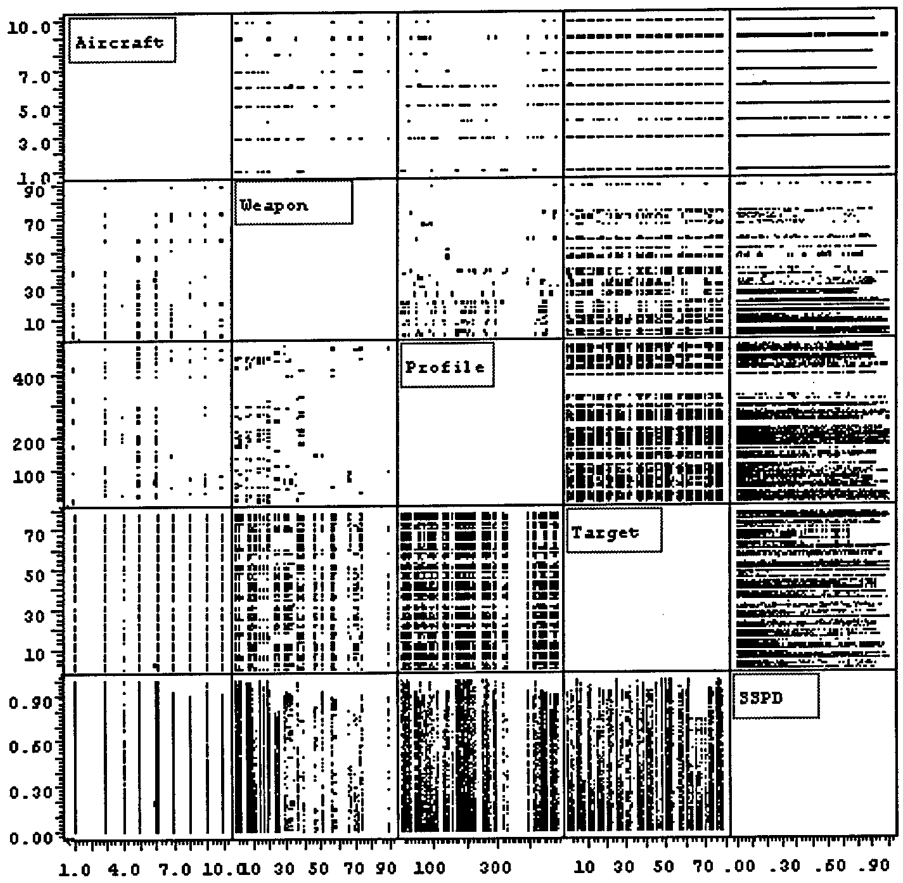


Figure 7 Scatterplot Matrix

Next the mean and standard deviation of the SSPD over each possible value taken by each feature (e.g. set aircraft equal to one and calculate the mean SSPD and standard deviation, then repeat for each aircraft) were then plotted. This was accomplished to determine if some structure could be created within the data. Figure 8 and figure 9 show the plots of the mean SSPD values and their standard deviations and demonstrates that creating structure would likely yield little benefit in network training due to the high degree of variance within each nominal value.

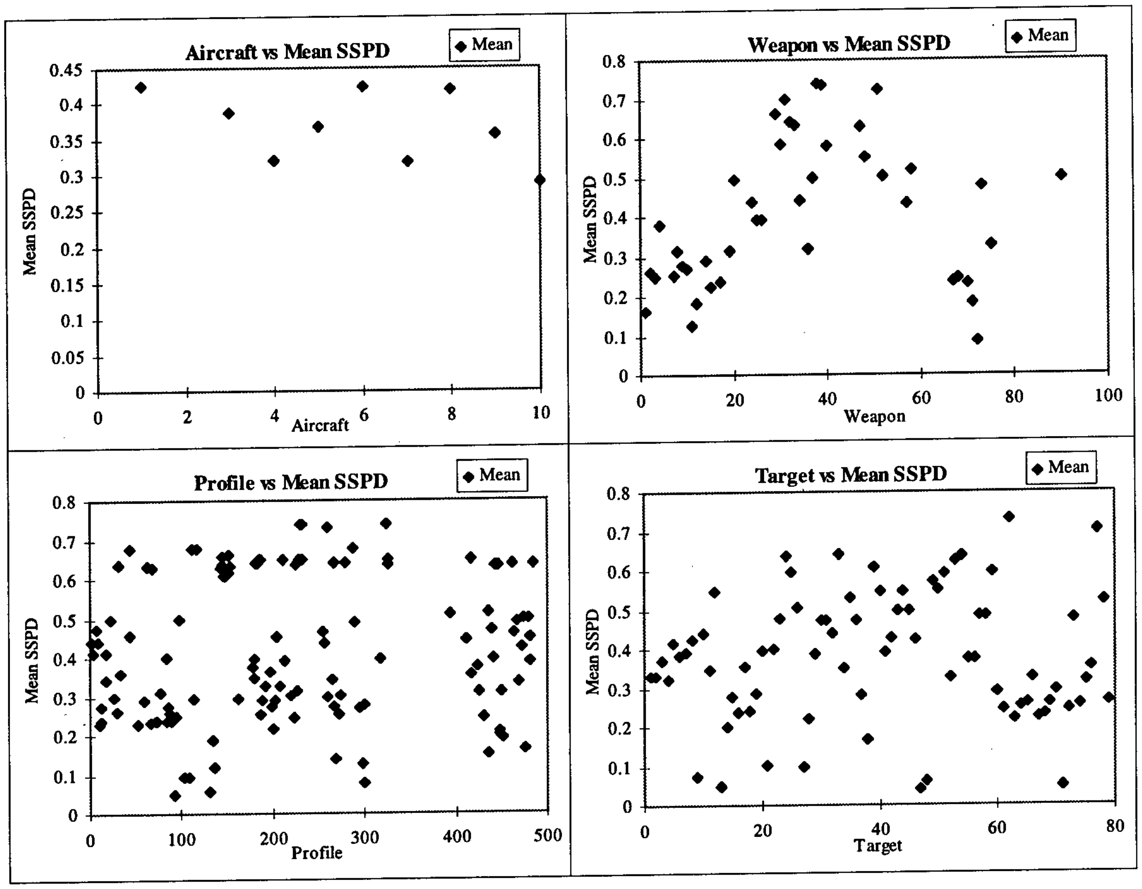


Figure 8. Feature vs. Mean SSPD

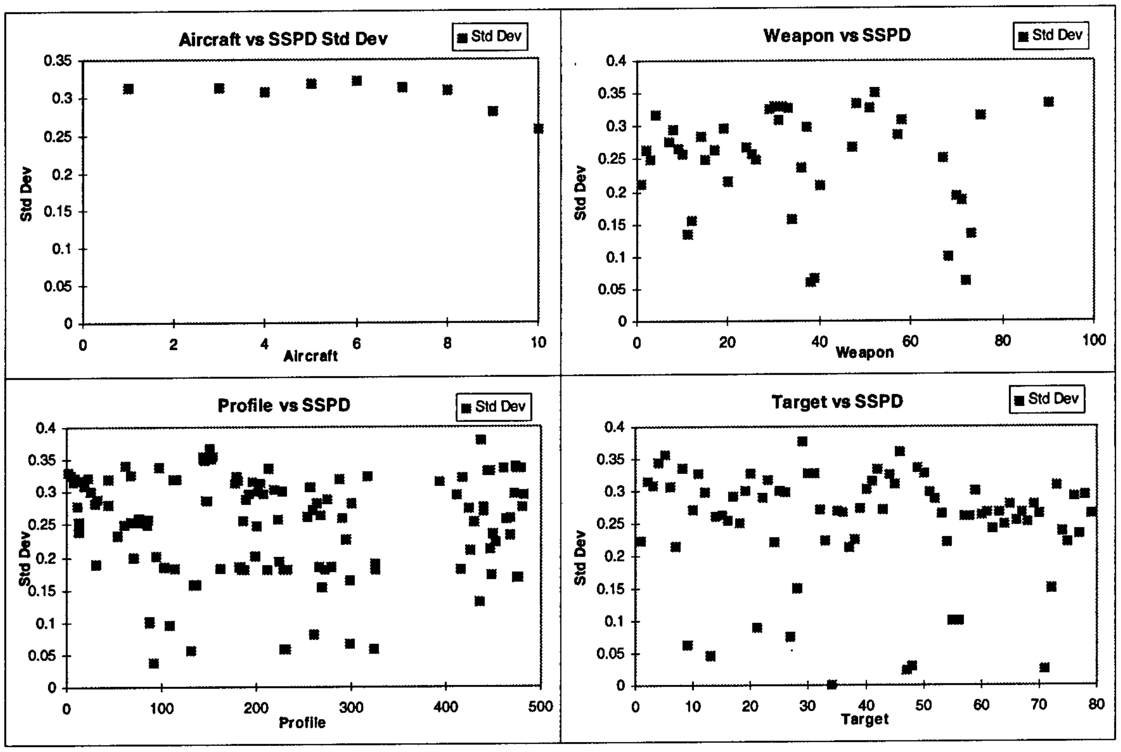


Figure 9. Feature vs. SSPD Std. Dev.

Once again various attempts to train the network were made by varying the number of hidden nodes from 10 to 100. The reduction of variables and exemplars, facilitated a significant improvement in the learning rate and enabled the use of up to 65 nodes before network training bogged down due to exhausting the available RAM. These efforts reduced the network RMSE slightly to 0.22 after 20000 epochs, but failed to get close to the desired 0.01.

3.1.2.2 Preprocessing the Data

According to Rogers, correctly preprocessing the data can speed learning (Rogers, Nov 1996). Two methods of preprocessing the data were implemented, normalizing the data (~N(0,1)) and converting each nominal value to a binary value. Normalizing the data initially improved the network learning rate, however the networks with normalized data soon lagged the networks that had no preprocessing and produced a less accurate RMSE of 0.24. Conversion of each nominal value to a binary string greatly increased the feature space from four to thirty one and likewise greatly increased the processor time required for each epoch. However, this method yielded a network with a significantly improved RMSE of 0.12 after 20000 epochs and 250 hours of processor time over a three week period. Intuitively, the improved accuracy makes sense due to the nominal nature of the input features.

3.2 Radial Basis Functions

At this point the need for a method of training a network that was less computationally intensive, thus allowing more nodes in the hidden layer, became apparent. RBF networks held promise for such a solution. As discussed in chapter 2, centering a RBF(node) on each exemplar in the data set, produces a function that is linear in the parameters (the weights) and can be solved efficiently using the least mean squares algorithm with zero error. However, this produces a network that requires significantly more memory than a look-up table for this application. Using Matlab's implementation of Chen's orthogonal least squares (OLS) algorithm, attempts were made to train to the desired accuracy with significantly less processor time than backpropagation (Demuth, 1992). The initial attempt to use this iterative forward regression technique on the entire data set using only the four features defined above (no preprocessing), once again exhausted the resources of the Ultra SPARC computer. Further attempts to train would require a significantly reduced data set. To verify the operation of the Matlab code, training was attempted on 1/200th subset of the original data chosen at random. By varying the spread constant (width) of the RBF, the desired error goal was reached, but required three nodes for every four exemplars. The size of the training set was iteratively increased in steps to1/20th of the original data set when the computer once again bogged down. As the training set was increased, the number of nodes required to reach the error goal held fairly constant at three nodes to four exemplars, thus failing to achieve the goal of a compact approximation. Though this technique successfully approximated a subset of the data, it would prove unsuitable for the entire data set due to processor limitations.

3.3 Divide and Conquer

At this point, neither backpropagation nor RBF networks provided an approximation of the data with the desired accuracy. Since the four features that span the SSPD output space are all nominal values, this facilitates reducing the feature space to three features while holding one feature value constant. Training a localized network for each value a feature takes within the set can greatly speed training due to the reduced feature space and the smaller subset of exemplars used for training. Once trained, these localized networks can be linked to produce a function approximation that spans the entire data set.

The first feature selected for this approach was aircraft type. This was selected because there were only nine possible values aircraft could take and therefore would require the training and linking of only nine networks. For the initial attempt using this methodology, an aircraft type for which there were relatively few exemplars (aircraft number nine, with 919 exemplars) was chosen. As expected, this network trained much faster on RBF networks and enabled the use of more nodes in the hidden layer using backpropagation, before the computer bogged down. The RBF network achieved the desired error goal but required two hidden layer nodes for every three exemplars. However, training on a larger set such as aircraft number three with 4288 exemplars overtaxed the computing capacity. Using this approach with a backpropagation network, showed greatly improved training and a significantly reduced trained network error. However, using 100 nodes in a backpropagation network produced a best RMSE result of 0.06.

The next feature selected for this technique was weapon type. This was chosen because it would require the next smallest number of trained networks to provide a complete function approximation. This feature assumed 43 possible values and would require the training and linking of a like number of networks. Due to lessons learned while holding aircraft type constant, a back propagation network was first trained. Again, the smaller subset of data produced quicker training and more accurate results. Training on these networks typically required eight to twelve hours of processor time. These networks achieved the desired RMSE goal of 0.01 while requiring approximately one hidden layer node to ten exemplars for each weapon type upon which a network was trained. Having achieved the desired accuracy and compactness with backpropagation, RBF networks were no longer attempted. This decision was made because RBF networks consistently required a much greater number of nodes to achieve a given error goal than backpropagation.

3.4 Statistical Networks - ModelQuest

ModelQuest has proven useful in a variety of applications such as stock prediction, flight controls and pattern recognition. Due to the relative ease of constructing and training a network using the ModelQuest software, as the features were reduced an attempt was made to approximate the function with this software essentially in parallel with the efforts using neural networks. As in the previous discussion on backpropagation, reduction of features greatly reduced training time. At each step, the ModelQuest software trained in under half the time required for backpropagation, but was never able to achieve accuracy results near those of backpropagation or RBF networks. The primary reason for this points back to the nature of the data. The software was designed to quickly numerically approximate a function given the assumption that numerical values have some real meaning.

The effort that produced the best results, though far from satisfactory, entailed holding the feature, weapon, constant as described in section 3.3. This produced a RMSE of 0.29. In fairness, ModelQuest was not designed to work well with data that consists entirely of nominal input values. However, at the beginning of this investigation, the nature of the features was not known. One suggestion given by the vendor to improve the performance entailed binary coding each feature. This was not attempted since it would expand the feature space from 4 to 261 features, thus increasing training time beyond reasonable bounds.

3.5 Statistical Approximation

In addition to the statistical methods discussed in section 3.1.2.1 for feature reduction, correlation analysis, least mean squares fitting and stepwise regression were performed on this data. Principal component analysis resulted in four principal components, each of which related was highly correlated to one of the four features selected, thus confirming the feature reduction already accomplished. Using the JMP software, least mean squares fitting produced first order model with 257 parameters and a RMSE of 0.19. The reason for the high number of parameters is because JMP treated each value assumed by each feature as a separate variable. Examination of higher order models up to fourth order using stepwise regression netted only a minor improvement, a RMSE of 0.185 while using over 1400 parameters. Neither of these techniques met the accuracy goal or the goal of a compact representation of the data.

3.6 Summarizing Function Approximation

All function approximation attempts to this point failed to produce a result that provides the desired output efficiently. However, achieving the desired accuracy using a backpropagation neural network demonstrates an ability to approximate even discontinuous data. The reduction of features improved performance for all techniques. In table 4, the best RMSE is listed for each methodology along with comments on relative requirements needed to apply each method.

Table 4. Results Summary

Methodology		Best RMSE	Computational Requirement	Practitioner Skill Required	Nodes or Processing Elements Required
Neural Networks					
	Backpropagation	0.01	Very High	High	Moderate
	Radial Basis Functions	***	High	High	Very High
ModelQuest					
	Statistical Networks	0.29	Moderate	Low	Moderate
Statistical Methods					
	Least Mean Squares	0.19	Low	Low	Moderate
	Stepwise Regression	0.185	Low	Moderate	Very High

Note: RBF's achieved the desired RMSE, but only on a subset of the data.

3.7 Aggregation with a Look-up Table

One of the underlying goals for function approximation was the creation of a compact representation of the data set. However due to the nature of the data, this goal was not met. Since MS Excel, Access and other spreadsheet and database applications provide efficient tools for looking up values within a data set, the decision was made to use a look-up table rather than a function approximation. MS Access was chosen to build an application tool due to its application building features and its ability to handle large data sets. Due to the desire for flexibility in the number of aircraft, weapons and targets to be aggregated at any given time, a weighted average of the look-up values was chosen for the purposes of aggregation.

Specifically, Access is used to filter all possible combinations of the input values supplied for aircraft type, weapon type, and target number to return a list of SSPD values. From this point, the aggregate SSPD may be calculated using one of two methods, either by feature or as a total weighted sum across the input features aircraft, weapon and target. In either case, the aggregate value is a weighted sum, with the sum of the weights equal to 1.0. Calculation of the weighted sum by feature begins by calculating the mean SSPD for each distinct value a feature takes using:

$$\bar{x}_j = \frac{1}{n}\sum_{i=1}^{n} x_i \text{ , } i = 1,\ldots,n, \tag{16}$$

where x_i are the best SSPD values returned by the look-up for a given weather condition, n is the number rows returned for a given feature value, and $\bar{x}_j$ is the mean SSPD for feature x when it takes value j, $j = 1,\ldots,m$, and m is the number of distinct values for feature x. The weighted SSPD for feature x is then taken using:

$$\overline{X} = \sum_{j=1}^{m} \overline{x}_j w_j, j = 1,\ldots,m, \tag{17}$$

where the w_j is the specified percentage of the force mix for a given $\overline{x}_j$. The w_j for the feature aircraft type are adjusted for differing weapon loads using:

$$w_j = \frac{w_s * loadout_i}{\sum_{j=1}^{m} w_s * loadout_i}, \tag{18}$$

where s is the specified portion of the aircraft mix and *loadout* is number of weapons carried by an aircraft. Calculation of the total weighted sum across all features follows a similar logic:

$$SSPD = \sum_{k=1}^{r} \sum_{l} x_k w_l, k = 1,\ldots,r, l = \{aircraft, weapon, t\arg et\}, \tag{19}$$

where r is the total number of values returned by filtering the SABSEL data, l designates the three features, aircraft, weapon and target. Again the weights are normalized such that:

$$\sum_{k=1}^{r} \sum_{l} w_l = 1.0. \tag{20}$$

A series of queries were built to implement the above aggregation in Access that meets the desires of CCCA. This application filters the SABSEL data in under thirty seconds on a Pentium 90 PC with zero error and requires an additional one to two minutes to complete the aggregation process. Data entry forms were created to make the process relatively easy, not requiring more than a few minutes to complete. A tutorial for this application is at Appendix D.

3.8 Summary

This chapter discussed the steps taken to build an accurate function approximation of the SABSEL data using neural networks, statistical networks and traditional statistical techniques. A discussion of the methods used for feature reduction was also included. Table 4 summarizes the results of the function approximation attempts, showing a relative comparison of the techniques. Finally, a brief discussion of the building of an aggregation tool using Access was provided.

IV. Conclusions/Recommendations

The purpose of this thesis was to produce a tool for the aggregation of SSPD value from the relatively large SABSEL database. After drawing some conclusions on the performance of each type of function approximation technique as discussed in chapter 3, this chapter will recommend areas of future study.

4.1 Conclusions

4.1.1 Data Reduction

All attempts at approximation of the SABSEL data were greatly affected by the nature of the data. Once feature reduction was accomplished, each method showed significant performance improvements both in terms of training time and accuracy. Any feature reduction that can be accomplished prior to training the networks will be highly beneficial.

4.1.2 Neural Networks

Both backpropagation networks and radial basis function networks can be highly computationally intense with a large data set. Splitting the data can provide benefits in terms of training time while providing a source of data for evaluation of the trained network. Due to the relatively uniform distribution of the data, neither backpropagation or RBF's proved simple. Only after breaking the data into small subsets corresponding to some nominal value, was a network trained to a suitably small error. This suggests that as processors get faster a network may be trained on a data set of this size and nature.

Statistical evaluation of the data provided great insight into the nature of the data and should be used along side neural network techniques.

4.1.3 Statistical Networks

The ModelQuest software package was unsuited for this task. The function approximation required for this research called for a scalpel, while the ModelQuest software provided a butcher knife. Though it does accomplish function approximation, it could not provide a highly accurate approximation using the SABSEL's nominal data . The uniform distribution of the data within the space and the accuracy requirement can account for the relatively poor performance of this piecewise approximation technique. To its advantage, this software was intuitive and relatively fast (compared to backpropagation).

4.1.4 Statistical Techniques

Traditional regression techniques for approximating this highly uniform data proved inadequate. Though simple to execute using the JMP software package(JMP,1995), both least mean squares regression and higher order stepwise regression failed to approach the accuracy requirement of this research. Again much of this failure can be attributed to the nature of the data.

4.1.5 Look-up vs. Function Approximation

Once feature reduction had been accomplished, the data set consisted of a set of features composed of nominal values. Though backpropagation could achieve the desired accuracy, there exist many applications that perform table look-up with much less effort and greater efficiency. Function approximation by any of the methods attempted is more suited to instances where there are continuous or at least ordinal values within the data set, rather than nominal values as was the case for the SABSEL data. The choice of using a look-up table or a function approximation for aggregation should be made depending on the nature of the data. In this case, the look-up table proved far superior to function approximation for aggregating the data.

4.2 Aggregation Tool

The suitability of the look-up table aggregation can best be summarized by CCCA. For their needs the SABSEL Aggregation tool:

"works very well and will allow us to improve the quality of the probabilities of kill we use in our combat models such as TACWAR. In addition it should reduce the time needed to perform an aggregation which will us to regularly update our data base every time we receive a new SABSEL data base. It also provides the mechanism to clearly define the aggregation assumptions used and to record those assumptions for future reference and repeated use." (Hebert, 1997)

4.3 Recommendations

Within the modeling and simulation community there exist many stochastic models where much data is available, yet there is no satisfactory means of mining the information from the data. For example, a multitude of data exists for many scenarios of the THUNDER model that has not been aggregated into tool that provides a meaningful way of predicting the results of the model. Neural networks, and statistical networks, once trained, can be used to provide an 80% answer quickly rather than requiring numerous model runs.

To accomplish such a mining of existing data the following is recommended:

1. Determine the nature of the data. Is the data primarily continuous, ordinal, or nominal? Use statistical techniques such as scatterplots, correlation analysis, and principal components analysis. Also examine various groupings of the data, searching for any group of nominal values that spans the data space.
2. Reduce the feature space. Using information gained in step 1, eliminate unnecessary features.
3. Depending on the nature of the data:
   a. Primarily continuous data: use statistical networks to achieve a ballpark type approximation (because of its relative ease of use), or neural networks if a more precise approximation desired.
   b. Primarily nominal data: use a spreadsheet or database program to provide a look-up that executes quickly and provides data for further aggregation.
   c. Mixed data: Train a network for each nominal value, and use a look-up as a pointer to each network, or train a network of networks as suggested by ModelQuest (ModelQuest, 1996:4.22).

4.4 Summary

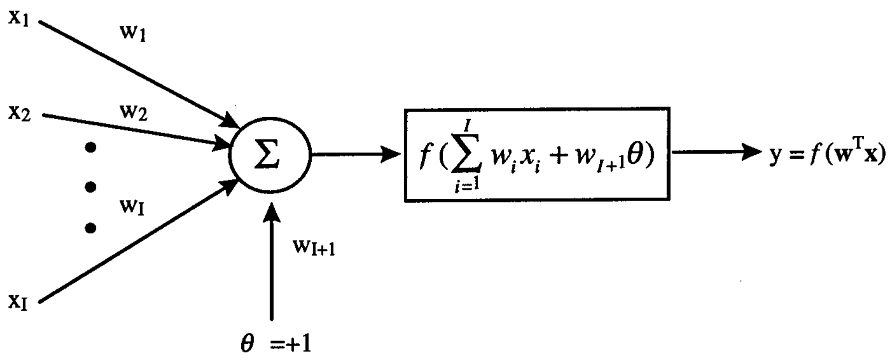
The SABSEL data primarily consists of nominal data features. Neural networks, Statistical networks and regression techniques were compared with a look-up table methodology on the basis of accuracy, speed and compactness (storage requirement). An Access based tool has been created that quickly filters the SABSEL data and implements a weighted sum algorithm to aggregate the SSPD values. A brief tutorial is provided at appendix D. This tool should greatly simplify the aggregation of SABSEL SSPD values for use in the TACWAR model.

Appendix A. Glossary of Neural Network Terms

- **Back-propagation** - a supervised learning algorithm for adjusting weights in a multilayer, feedforward neural network that uses gradient descent techniques to minimize network output error.
- **Bias** - the bias, θ, provides a means of shifting the intercept of the transfer function. Its input value is set at 1.0 and its associated weight shifts the intercept.
- **Epoch** - the presentation of all exemplars in the data set being used to train the network or perceptron one time..
- **Exemplar** - an **input** data **vector** consisting of a unique set of feature values, $x_1, x_2, \ldots, x_I$. An exemplar in neural network terminology is the same thing as a sample point in statistical terminology. Essentially this is one row of the data set.
- **Feature** - an independent variable, x_i, which provides information useful for approximating a function or distinguishing classes.
- **Feedforward Network** - a collection of perceptrons whose connections exclusively feed inputs from lower to higher layers; in contrast to a feedback network, a feedforward network operates only until its inputs propagate to its output layer (DARPA, 1988).
- **Hidden Layers** - those processing elements in a multilayer neural network which are neither the input layer or output layer.
- **Multilayer Perceptron (MLP)**- a feedforward multilayer perceptron that is fully connected(i.e. each input from a lower layer is connected to each perceptron in the next higher layer). A typical MLP has an input layer, one or more hidden layers and

an output layer. This type network is usually trained through back-propagation. See Figure A-1 Multilayer Perceptron .

- **Perceptron** - first proposed by Rosenblatt (Rosenblatt, 1959), the single perceptron , Figure A-1, performs a transformation on the weighted sum of the input exemplar. Perceptrons are also known as neurons, or nodes, or processing elements.



x_1
w_1
x_2
w_2
w_I
x_I
Σ
$f(\sum_{i=1}^{I} w_i x_i + w_{I+1}\theta)$
$y = f(\mathbf{w}^T\mathbf{x})$
w_{I+1}
$\theta = +1$

Figure A-1. Perceptron

- **Supervised Training** - a method of training a neural network that provides the desired output to the network as a standard from which to measure network output error. An alternative technique is unsupervised training where the network "self organizes" or clusters the data during the training process.

- **Transformation Function** - the perceptron maps the weighted sum of its inputs onto its output through the transformation function, $f(\bullet)$ (also known as a transfer function).

- **Weight** - each input into a perceptron is weighted, $w_1, w_2, \ldots, w_I$. The weights are adjusted through training to achieve the desired output.

Appendix B. Chen's Orthogonal Least Squares Basis Function Algorithm

The RBF network implements a mapping $f_r : \mathbf{R}^n \rightarrow \mathbf{R}$ as follows:

$$f_r(\mathbf{x}) = \theta + \sum_{i=1}^{I} w_i \phi(\|\mathbf{x} - \mathbf{c}_i\|), \tag{21}$$

where $\mathbf{x} \in \mathbf{R}^n$ is the exemplar(input vector), $\phi(\bullet)$ is a given RBF, θ is the bias (intercept), w_i denotes the weights (parameters), c_i are the RBF centers, $\|\bullet\|$ denotes the Euclidean norm, and I is the number of centers. Assuming that the centers c_i and the functional form $\phi(\bullet)$ are fixed, weights w_i can be determined using linear least squares (LS). According to Chen, the choice of the type of RBF $\phi(\bullet)$ is not significant. The Gaussian function is

$$\phi(v) = \exp(-v^2 / \beta^2), \tag{22}$$

where v is the Euclidean distance from the center, and β is a preset spread constant. (Chen , 1991:303). A larger β results in smoother function, while too large a value of β can result in an ill conditioned matrix(Demuth, 1992).

In his presentation of the orthogonal least squares (OLS) algorithm, Chen views the RBF network as a special case of the linear regression model

$$d(t) = \sum_{i=1}^{M} p_i(t)\theta_i + \varepsilon(t), \tag{23}$$

where d(t) is the desired output, θ_i are the parameters (weights), and $p_i(t)$ are the regressors which are some fixed functions of $\mathbf{x(t)}$:

$$p_i(t) = p_i(\mathbf{x(t)}). \tag{24}$$

Equation 20 can be rewritten as:

$$\mathbf{d} = \mathbf{P}\Theta + \mathbf{E} \tag{25}$$

where

$$\mathbf{d} = [d(1) \ldots d(N)]^T, \tag{26}$$

$$\mathbf{P} = [\mathbf{p}_1 \ldots \mathbf{p}_M],\ \mathbf{p}_i = [p_i(1) \ldots p_i(N)]^T,\ 1 \le i \le M, \tag{27}$$

where M is the number of RBF centers,

$$\Theta = [\theta_1 \ldots \theta_M]^T, \tag{28}$$

$$E = [\varepsilon(1) \ldots \varepsilon(N)]^T. \tag{29}$$

Chen's OLS method transforms the set of $\mathbf{p}_i$ into a set of orthogonal basis vectors. This transformation makes it possible to calculate the individual contribution to the desired output from each basis vector. In other words, this transformation provides a way to rank each RBF center according to its contribution to achieving the desired accuracy.

Decomposing the regression matrix **P** results in:

$$\mathbf{P} = \mathbf{WA}, \tag{30}$$

where **A** is an $M \times M$ triangular matrix with 1's on the diagonal and 0's below the diagonal, and W is an $N \times M$ matrix with orthogonal columns w_i such that

$$\mathbf{W}^T\mathbf{W} = \mathbf{H} \tag{31}$$

where **H** is diagonal with elements h_i:

$$h_i = \mathbf{w}_i{}^T\mathbf{w}_i = \sum_{i=1}^{N} w_i(t)w_i(t),\ 1 \le i \le M. \tag{32}$$

The orthogonal basis vectors $\mathbf{w}_i$ span the same space as the set of $\mathbf{p}_i$ an can be written as

$$\mathbf{d} = \mathbf{Wg} + E. \tag{33}$$

The OLS solution can then be given by

$$\hat{\mathbf{g}} = \mathbf{H}^{-1}\mathbf{W}^T\mathbf{d} \tag{34}$$

or

$$\hat{g}_i = \mathbf{w}_i^T\mathbf{d} / (\mathbf{w}_i^T\mathbf{w}_i),\ 1 \le i \le M. \tag{35}$$

The OLS algorithm facilitates the selection of a subset of RBF centers from a large set of data. By equating $\mathbf{w}_i$ to $\mathbf{p}_i$, Chen proceeds to derive an error reduction ratio due to $\mathbf{w}_i$

$$[err]_i = g_i^2\mathbf{w}_i^T\mathbf{w}_i / (\mathbf{d}^T\mathbf{d}),\ 1 \le i \le M. \tag{36}$$

This error ratio offers a simple method for selecting a set of significant regressors using forward regression. Finally, Chen summarizes this procedure as follows:

1. For $1 \le i \le M$, compute

$$\left.\begin{aligned} &\mathbf{w}_1^{(i)} = \mathbf{p}_i \\ &g_1^{(i)} = (\mathbf{w}_1^{(i)})^T\mathbf{d} / ((\mathbf{w}_1^{(i)})^T\mathbf{w}_1^{(i)}) \\ &[err]_1^{(i)} = (g_1^{(i)})^2(\mathbf{w}_1^{(i)})^T\mathbf{w}_1^{(i)} / \mathbf{d}^T\mathbf{d} \end{aligned}\right\}. \tag{37}$$

Find

$$[err]_1^{(i_1)} = \max\{[err]_1^{(i)},\ 1 \le i \le M\} \tag{38}$$

and select

$$\mathbf{w}_1 = \mathbf{w}_1^{(i_1)} = \mathbf{p}_{i_1}. \tag{39}$$

2. At the kth step where $k \ge 2$, for $1 \le i \le M$, $i \ne i_1, \ldots, i \ne i_{k-1}$, compute

$$\left.\begin{aligned} &\alpha_{jk}^{(i)} = \mathbf{w}_j^T\mathbf{p}_i / (\mathbf{w}_j^T\mathbf{w}_j), 1 \le j < k \\ &\mathbf{w}_k^{(i)} = \mathbf{p}_i - \sum_{j=1}^{k-1}\alpha_{jk}^{(i)}\mathbf{w}_j \\ &g_k^{(i)} = (\mathbf{w}_k^{(i)})^T\mathbf{d} / ((\mathbf{w}_k^{(i)})^T\mathbf{w}_k^{(i)}) \\ &[err]_k^{(i)} = (g_k^{(i)})^2\mathbf{w}_k^{(i)})^T\mathbf{w}_k^{(i)} / \mathbf{d}^T\mathbf{d} \end{aligned}\right\}. \tag{40}$$

Find

$$[err]_1^{(i_1)} = \max\{[err]_1^{(i)},\ 1 \le i \le M, \quad i \ne i_1, \ldots, i \ne i_{k-1}, \tag{41}$$

and select

$$\mathbf{w}_k = \mathbf{w}_k^{(i_k)} = \mathbf{p}_{ik} - \sum_{j=1}^{k-1} \alpha_{jk} \mathbf{w}_j \tag{42}$$

$$\text{where } \alpha_{jk} = \alpha_{jk}^{i_k}, 1 \le j < k. \tag{43}$$

3. Stop when

$$1 - \sum_{j=1}^{M_s} [err]_j < \rho \tag{44}$$

where M_s is the number of significant regressors and ρ, $0 < \rho < 1$, is the chosen tolerance(Chen, 1991:303-305).

Example Matlab Code for Backpropagation (Levenberg-Marquardt Algorithm)

```
% 2 Dec 96.
% Maj Parker.
% After performing analysis on the data and reducing the features, this
% trial attempts to train on only those exemplars where the feature 'aircraft' has a
% value of 1, using LM.

% initialize parameters
clear; close all;

%  Load the SABSEL data, selecting the desired features

load  aircraft1.mat;
P = aircraft1(:,5:7)';

%    T defines the associated 1-element targets (column vectors):

T = aircraft1(:,4)';
clear aircraft1;

% Normalize the columns of the input matrix P.

P = normc(P);

%    PLOTTING THE DATA POINTS
%    ======================
%    Here the data points are plotted:

figure(1);
plot(P,T,'+');
title('Training Vectors');
xlabel('Input Vector P');
ylabel('Target Vector T');

%    The function the neural network learns must pass through
%    these data points.

%    DESIGN THE NETWORK
%    ==================
%    A two-layer TANSIG/PURELIN network will be trained using 100  hidden nodes.

S1 = 100;

%    INITFF is used to initialize the weights and biases for
%    the TANSIG/PURELIN network.

[w1,b1,w2,b2] = initff(P,S1,'tansig',T,'logsig');

%    TRAINING THE NETWORK
%    ==================
%    TRAINLM uses backpropagation to train feed-forward networks.
```

```
df  = 25;            % Frequency of progress displays (in epochs).
me  = 20000;        % Maximum number of epochs to train.
eg  = 0.1;          % Sum-squared error goal.
min_grad  = 0.0001;         % Minimum gradient
mu = 0.001;         % Initial value for mu.
mu_inc = 10;        % Multiplier for increasing mu.
mu_dec = 0.1;       % Multiplier for decreasing mu.
maxmu = 1e10;  % Maximum value for mu.

tp = [df me eg min_grad mu mu_inc mu_dec maxmu];

[w1,b1,w2,b2,ep,tr] = trainlm(w1,b1,'tansig',w2,b2,'logsig',P,T,tp);

%   PLOTTING THE ERROR CURVE
%   ======================
%   Here the errors are plotted with respect to training epochs:

figure;ploterr(tr,eg);
figure;
plot(P,T,'+');
title('Training Vectors');
xlabel('Input Vector P');
ylabel('Target Vector T');

% Save weights and input parameters upon completion.

clear P T;
save res100ac1;

end;
```

Example Matlab Code for Radial Basis Functions

```
% 4 Nov 96.
% Maj Parker.
% After performing analysis on the data and reducing the features, this
% trial attempts to train on only those exemplars where the feature 'aircraft' has a
% value of 1, using radial basis functions.

% initialize parameters
clear;  close all;

load  aircraft1.mat;
[n,m]=size(aircraft1);

%   P contains the normalized training vectors: Aircraft, Weapon, Profile, and Target.

temp = aircraft9(:,1:3);
temp = normc(temp);
P=temp';

%   T defines the associated 1-element targets (SSPD):
```

```
T=aircraft1(:,4)';
clear aircraft1 temp;


%  A two-layer radial basis network will be trained.
%  TRAINING THE NETWORK
%  ===================
%  SOLVERB trains a radial basis network using the minimum number of neurons
%  to meet the error goal.

df = 1;          % Frequency of progress displays (in epochs).
me = 500;        % Maximum number of to include in network.
eg = .20;        % Sum-squared error goal.
spread = 0.3;  % Spread for hidden layer transfer function.

tp = [df me eg spread];

[w1,b1,w2,b2,nr,dr] = solverb(P,T,tp);

% Save weights and input parameters upon completion.

clear P T;
save res100ac1;

end;
```

Appendix D. SABSEL Aggregator Tutorial

Introduction

The SABSEL Aggregator is designed to calculate a weighted average SSPD value from the SABSEL data set. Given the weather minimums, a set of aircraft, a set of weapons, and a set of targets, this tool filters the SABSEL data for SSPD values for all possible combinations of the aircraft, weapons and targets. These SSPD values are then aggregated into a single value based on the mix. For instance, if a particular aircraft makes up only 20% of the mix, it's influence on the aggregated SSPD is adjusted appropriately. Additionally, if an input combination does not exist within the SABSEL data, the mix of the other inputs are adjusted automatically. Finally, the specified mix for aircraft are adjusted for the number of weapons carried (e.g. an aircraft that can carry four of a particular weapon carries twice the weight of one that can carry only two). The following is a simple description of how to run the aggregator in MS Access. More detailed discussion of the algorithm is provided in chapter 3.

Instructions

1. Open Access and the sab.mdb file.
2. Select the forms tab and double click on the Aggregator Input form as shown in figure D-1.
3. The Data Input form (figure D-2) has four buttons for input. Click the Aircraft Input, Weapon Input, or Target Input button to view the data entry form (figure D-3). Drop down menus are available for legal values for aircraft type, weapon type and target number. Use care to ensure the sum of

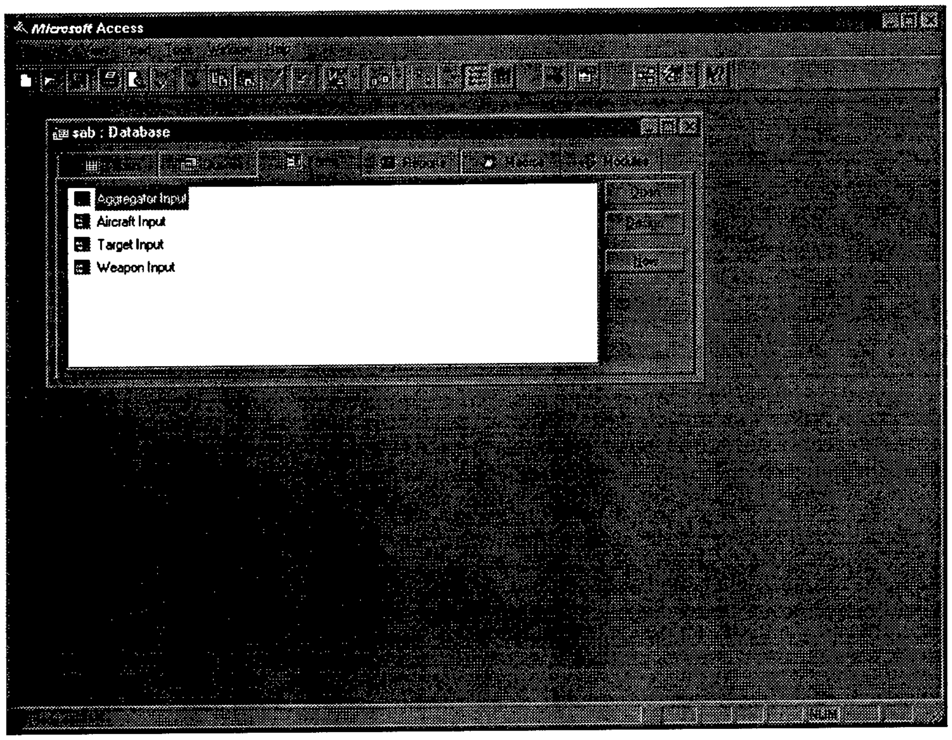
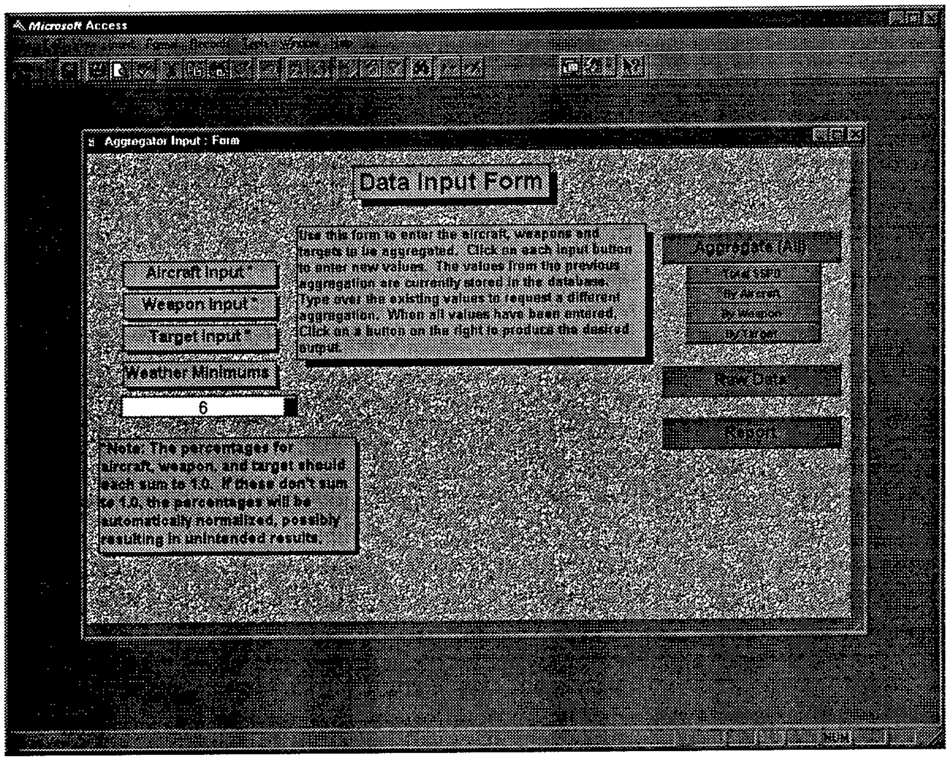


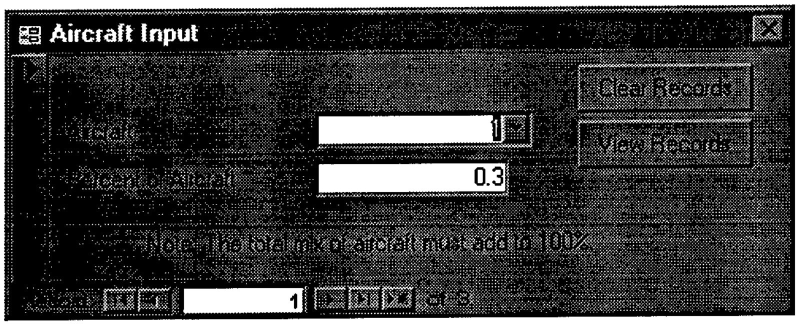
Figure D-1. Select Form



Use this form to enter the aircraft, weapons and targets to be aggregated. Click on each input button to enter new values. The values from the previous aggregation are currently stored in the database. Type over the existing values to request a different aggregation. When all values have been entered, Click on a button on the right to produce the desired output.

*Note: The percentages for aircraft, weapon, and target should each sum to 1.0. If these don't sum to 1.0, the percentages will be automatically normalized, possibly resulting in unintended results.

Figure D-2. Data Input Form



Aircraft: 1

Percent of Aircraft: 0.3

Note: The total mix of aircraft must add to 100%

Record: 1 of 3

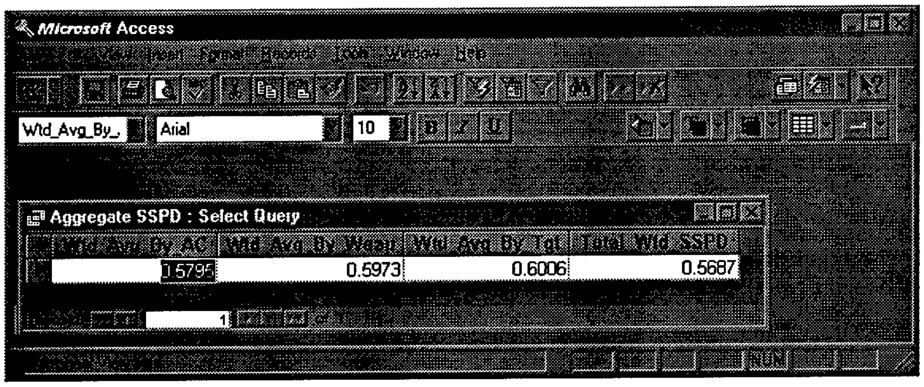
Figure D-3. Aircraft Entry Form

the mix input equals 1.0. If this sum does not equal 1.0, the program will automatically normalize the input mix. Each input form will initially display inputs stored previously. Type over or delete these values as desired.

4. Once the aircraft, weapons and targets have been input, specify the prevailing weather minimums, by using the drop down menu under "Weather Minimums." If the weather code is known, enter it directly.

5. Once the inputs are complete, this form provides three choices.

   a. To review the output, click the "Aggregate" button. This will present the weighted mean SSPD by aircraft, by weapon, and by target, along with a total aggregated value for SSPD, called "Total_Wtd_SSPD" (see figure D-4. Aggregate Button Output). The aggregator first selects the best SSPD for a given weather condition, aircraft, weapon and target from the SABSEL data. From this initial selection, the "Total_Wtd_SSPD" value is taken as a total weighted sum of all possible combinations, with the weights derived from the specified mix, and adjusted to sum to 1.0. **Normally, the "Total_Wtd_SSPD" value is the desired output.** However, there may be situations where one the type of aircraft,



Wtd Avg By AC	Wtd Avg By Wpn	Wtd Avg By Tgt	Total Wtd SSPD
0.5795	0.5973	0.6006	0.5687

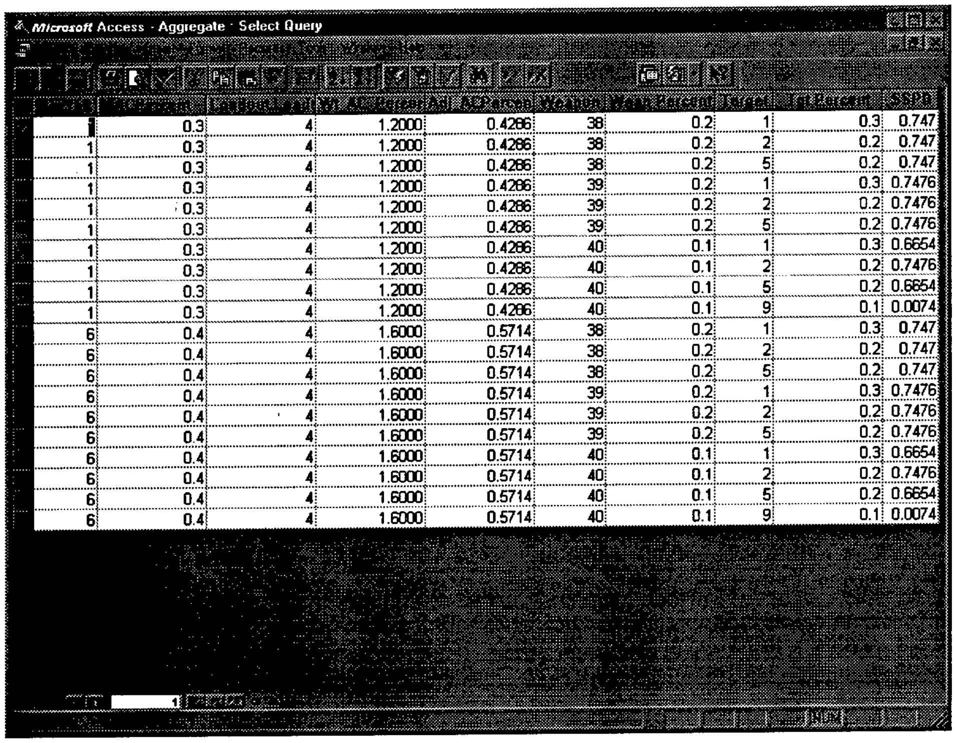
Figure D-4. Aggregate Button Output

weapon or target is the overriding factor. **The "By Aircraft", "By Weapon", and "By Target" buttons provide weighted sum SSPD values that are adjusted only for the named input**, and disregard the mix input values of the other inputs.

b. To view the raw data as extracted from the SABSEL data without any aggregation, click on "Raw Data." Figure D-5 shows an example of this output.

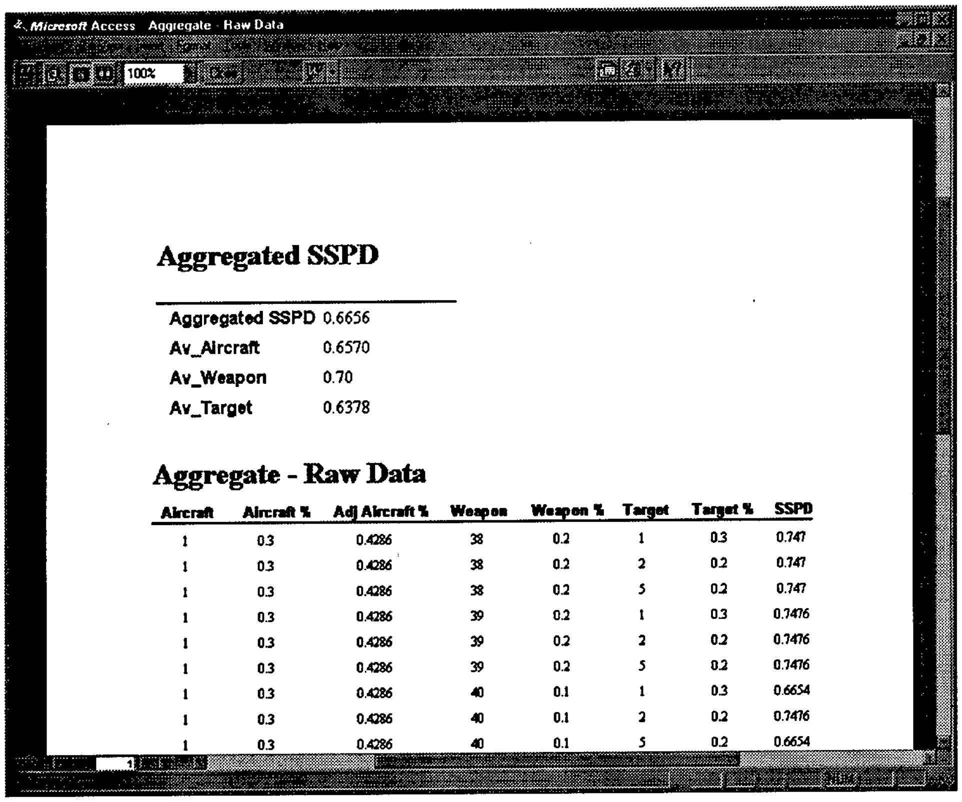
c. To get a report showing both the raw data and the aggregated SSPD values, click on "Report." Figure D-6 shows an example of this output. This option takes the longest to execute (approximately 2-3 minutes on a Pentium 90).

NOTE: Data from any of the above output can be transferred to and from MS Excel and Word using the cut and paste commands.


Microsoft Access - Aggregate - Select Query

[illegible]	[illegible]	[illegible]	Wt AC Percen Adj	AC Percen	Weapon	Weap Percent	Target	Tgt Percent	SSPD
1	0.3	4	1.2000	0.4286	38	0.2	1	0.3	0.747
1	0.3	4	1.2000	0.4286	38	0.2	2	0.2	0.747
1	0.3	4	1.2000	0.4286	38	0.2	5	0.2	0.747
1	0.3	4	1.2000	0.4286	39	0.2	1	0.3	0.7476
1	0.3	4	1.2000	0.4286	39	0.2	2	0.2	0.7476
1	0.3	4	1.2000	0.4286	39	0.2	5	0.2	0.7476
1	0.3	4	1.2000	0.4286	40	0.1	1	0.3	0.6654
1	0.3	4	1.2000	0.4286	40	0.1	2	0.2	0.7476
1	0.3	4	1.2000	0.4286	40	0.1	5	0.2	0.6654
1	0.3	4	1.2000	0.4286	40	0.1	9	0.1	0.0074
6	0.4	4	1.6000	0.5714	38	0.2	1	0.3	0.747
6	0.4	4	1.6000	0.5714	38	0.2	2	0.2	0.747
6	0.4	4	1.6000	0.5714	38	0.2	5	0.2	0.747
6	0.4	4	1.6000	0.5714	39	0.2	1	0.3	0.7476
6	0.4	4	1.6000	0.5714	39	0.2	2	0.2	0.7476
6	0.4	4	1.6000	0.5714	39	0.2	5	0.2	0.7476
6	0.4	4	1.6000	0.5714	40	0.1	1	0.3	0.6654
6	0.4	4	1.6000	0.5714	40	0.1	2	0.2	0.7476
6	0.4	4	1.6000	0.5714	40	0.1	5	0.2	0.6654
6	0.4	4	1.6000	0.5714	40	0.1	9	0.1	0.0074

Figure D-5. Raw Data Output


Microsoft Access - Aggregate - Raw Data

Aggregated SSPD

Aggregated SSPD	0.6656
Av_Aircraft	0.6570
Av_Weapon	0.70
Av_Target	0.6378

Aggregate - Raw Data

Aircraft	Aircraft %	Adj Aircraft %	Weapon	Weapon %	Target	Target %	SSPD
1	0.3	0.4286	38	0.2	1	0.3	0.747
1	0.3	0.4286	38	0.2	2	0.2	0.747
1	0.3	0.4286	38	0.2	5	0.2	0.747
1	0.3	0.4286	39	0.2	1	0.3	0.7476
1	0.3	0.4286	39	0.2	2	0.2	0.7476
1	0.3	0.4286	39	0.2	5	0.2	0.7476
1	0.3	0.4286	40	0.1	1	0.3	0.6654
1	0.3	0.4286	40	0.1	2	0.2	0.7476
1	0.3	0.4286	40	0.1	5	0.2	0.6654

Figure D-6. Report Output

Bibliography

Barron, A. R., "Predicted Square Error: A Criterion For Automatic Model Selection," Self-Organizing Methods in Modeling: GMDH Type Algorithms, edited by S. J. Farlow, New York: Marcel-Dekker, Inc., 1984

Bauer, Kenneth,. Class handout, OPER 685, Applied Multivariate Analysis, School of Engineering, Air Force Institute of Technology, Wright-Patterson AFB OH, September 1996.

Chen, S. *et al*, "Non-linear Systems Identification Using Radial Basis Functions," International Journal of Systems Science, vol. 21, no 1, pp. 2513-2539, 1990.

Cybenko, G. "Approximations by Superpositions of Sigmoidal Functions," Mathematics of Controls, Signals, and Systems, vol. 2, no. 4, pp. 303-314 ,1989

Demuth, Howard. Neural Network Toolbox for use with MATLAB: user's guide, Natick, Massachusetts: Mathworks, Inc., 1992.

Defense Advanced Research Projects Agency (DARPA), "Neural Network Study," AFCEA International Press, Fairfax VA, November 1988.

Foulk, John. Analysis officer, Centcom Combat Analysis Group, HQ Central Command, Patrick AFB, FL. Telephone Interview. 4 September 1996.

Hebert, Joseph. Analysis officer, Centcom Combat Analysis Group, HQ Central Command, Patrick AFB, FL. Telephone Interview. 4 March 1997.

JMP®. Version 3.1, IBM, 8MB, disk. Computer software. SAS Institute Inc., Cary, NC, 1995.

ModelQuest™ . Version 4.0, IBM, 15MB, disk. Computer software. AbTech Corporation, Charlottesville, VA, 1996.

Nelson, Marilyn McCord and W. T. Illingworth, A Practical Guide to Neural Nets. Reading, MA: Addison-Wesley Publishing Company, Inc., 1991.

Neter, John, *et al*, Applied Statistical Linear Models, Fourth Edition. Chicago, Times Mirror Higher Education Group, Inc., 1996.

Rogers, Steven K. and others, An Introduction to Biological and Artificial Neural Networks for Pattern Recognition. Boston, Society of Photooptical instrumentation Engineers Optical Engineering Press, 1991.

Rogers, Steven K. Introduction to Perceptrons: Advanced Topics in Neural Networks. Air Force Institute of Technology, Wright-Patterson AFB, OH, April 1996.

Rogers , Steven K. Class notes, ENG 699, Introduction to Biological and Artificial Neural Networks, School of Engineering, Air Force Institute of Technology, Wright-Patterson AFB OH, November 1996.

Rosenblatt, R., Principles of Neurodynamics. Spartan Books, New York, 1959.

REPORT DOCUMENTATION PAGE	*Form Approved* *OMB No. 0704-0188*

1. AGENCY USE ONLY *(Leave blank)*	2. REPORT DATE February 1997	3. REPORT TYPE AND DATES COVERED Master's Thesis

4. TITLE AND SUBTITLE Experiments in Aggregating Air Ordinance Effectiveness Data for the TACWAR Model	5. FUNDING NUMBERS
6. AUTHOR(S) James E. Parker, Major, USAF	
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Air Force Institute of Technology/ENS 2750 P Street Wright-Patterson AFB, Ohio 45433-7765	8. PERFORMING ORGANIZATION REPORT NUMBER AFIT/GOA/ENS/97M-12
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES) HQ ACC/XP-SAS 204 Dodd Blvd., Ste. 202 Langley AFB, VA 23665-2778	10. SPONSORING/MONITORING AGENCY REPORT NUMBER
11. SUPPLEMENTARY NOTES Approved for Public Release; Distribution is Unlimited	
12a. DISTRIBUTION/AVAILABILITY STATEMENT	12b. DISTRIBUTION CODE

13. ABSTRACT *(Maximum 200 words)*

An interactive MS Access™ based application that aggregates the output of the SABSEL model for input into the TACWAR model is developed. The application was developed following efforts to create a functional approximation of the SABSEL data using neural networks, statistical networks, and traditional statistical techniques. These approximations were compared to a look-up table methodology on the basis of accuracy, (RMSE < 0.01), speed (runs in minutes) and compactness (storage requirement). The method best satisfying these criteria, the look-up table, was used as the basis from which the application was constructed. Results from the comparison of the function approximation techniques give insight as to the strengths and weaknesses of each technique.

14. SUBJECT TERMS Aggregation, Nueral Networks, Radial Basis Functions, Backpropagation, Statistical Networks, SABSEL			15. NUMBER OF PAGES 16. PRICE CODE
17. SECURITY CLASSIFICATION OF REPORT Unclassified	18. SECURITY CLASSIFICATION OF THIS PAGE Unclassified	19. SECURITY CLASSIFICATION OF ABSTRACT Unclassified	20. LIMITATION OF ABSTRACT UL

GENERAL INSTRUCTIONS FOR COMPLETING SF 298

The Report Documentation Page (RDP) is used in announcing and cataloging reports. It is important that this information be consistent with the rest of the report, particularly the cover and title page. Instructions for filling in each block of the form follow. It is important to ***stay within the lines*** to meet ***optical scanning requirements***.

Block 1. Agency Use Only *(Leave blank)*.

Block 2. Report Date. Full publication date including day, month, and year, if available (e.g. 1 Jan 88). Must cite at least the year.

Block 3. Type of Report and Dates Covered. State whether report is interim, final, etc. If applicable, enter inclusive report dates (e.g. 10 Jun 87 - 30 Jun 88).

Block 4. Title and Subtitle. A title is taken from the part of the report that provides the most meaningful and complete information. When a report is prepared in more than one volume, repeat the primary title, add volume number, and include subtitle for the specific volume. On classified documents enter the title classification in parentheses.

Block 5. Funding Numbers. To include contract and grant numbers; may include program element number(s), project number(s), task number(s), and work unit number(s). Use the following labels:

C -	Contract	PR -	Project
G -	Grant	TA -	Task
PE -	Program Element	WU -	Work Unit Accession No.

Block 6. Author(s). Name(s) of person(s) responsible for writing the report, performing the research, or credited with the content of the report. If editor or compiler, this should follow the name(s).

Block 7. Performing Organization Name(s) and Address(es). Self-explanatory.

Block 8. Performing Organization Report Number. Enter the unique alphanumeric report number(s) assigned by the organization performing the report.

Block 9. Sponsoring/Monitoring Agency Name(s) and Address(es). Self-explanatory.

Block 10. Sponsoring/Monitoring Agency Report Number. *(If known)*

Block 11. Supplementary Notes. Enter information not included elsewhere such as: Prepared in cooperation with...; Trans. of...; To be published in.... When a report is revised, include a statement whether the new report supersedes or supplements the older report.

Block 12a. Distribution/Availability Statement. Denotes public availability or limitations. Cite any availability to the public. Enter additional limitations or special markings in all capitals (e.g. NOFORN, REL, ITAR).

DOD -	See DoDD 5230.24, "Distribution Statements on Technical Documents."
DOE -	See authorities.
NASA -	See Handbook NHB 2200.2.
NTIS -	Leave blank.

Block 12b. Distribution Code.

DOD -	Leave blank.
DOE -	Enter DOE distribution categories from the Standard Distribution for Unclassified Scientific and Technical Reports.
NASA -	Leave blank.
NTIS -	Leave blank.

Block 13. Abstract. Include a brief *(Maximum 200 words)* factual summary of the most significant information contained in the report.

Block 14. Subject Terms. Keywords or phrases identifying major subjects in the report.

Block 15. Number of Pages. Enter the total number of pages.

Block 16. Price Code. Enter appropriate price code *(NTIS only)*.

Blocks 17. - 19. Security Classifications. Self-explanatory. Enter U.S. Security Classification in accordance with U.S. Security Regulations (i.e., UNCLASSIFIED). If form contains classified information, stamp classification on the top and bottom of the page.

Block 20. Limitation of Abstract. This block must be completed to assign a limitation to the abstract. Enter either UL (unlimited) or SAR (same as report). An entry in this block is necessary if the abstract is to be limited. If blank, the abstract is assumed to be unlimited.